
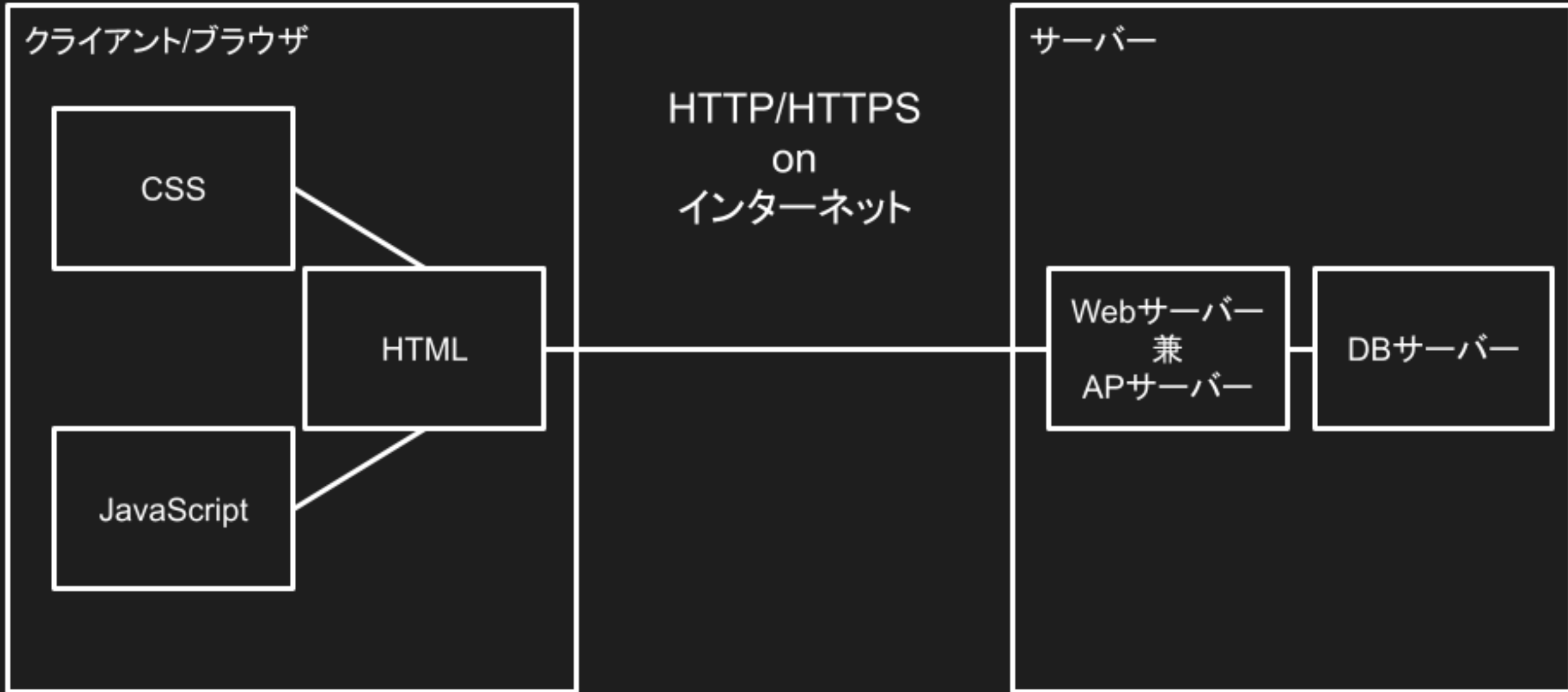


Webアプリケーション入門 🚀

JavaScriptの基本

はじめに：今日やること

- 今日のゴール: 
 - JavaScriptとは何かを知る
 - JavaScriptの基本的な書き方を知る
 - JavaScriptを使ってWebページに機能をつける



JavaScriptってなに？ 🤔

一言でいうと…

ウェブサイトに「動き」や「対話」を加える
ためのプログラミング言語です。

ウェブサイトをもっと面白く、便利にします！ ✨



ウェブサイトの3つの要素 🏠

ウェブサイトは主にこの3つでできています。

- **HTML:** 家の骨組み (文章、画像など)
 - 例: 柱、壁、部屋の配置
- **CSS:** 家の見た目 (色、デザイン、配置)
 - 例: 壁紙の色、家具のデザイン
- **JavaScript:** 家の機能・設備 (動き、対話)
 - 例: 電気、水道、自動ドア

➡ JavaScriptは「機能」を追加する役割です！💡




JavaScriptは何ができるの？ (1) 🚀

動きをつける

- 写真が自動で切り替わるスライドショー 🖼️
- マウスを乗せると色が変わるボタン ✨
- クリックで現れるドロップダウンメニュー 📄

JavaScriptは何ができるの？ (2)

ユーザーと対話する


- フォーム入力の間違いをチェック 
 - 例: "メールアドレスの形式が違います！"
- ボタンクリックでメッセージ表示 
 - 例: "送信しました！"
- 簡単な自動計算 
 - 例: 商品の合計金額

JavaScriptは何ができるの？ (3) ✨

ページをスムーズに見せる (非同期通信)

- 地図をグリグリ動かせる (Google マップなど) 🌐
- 「いいね！」がページ移動なしで増える ❤️
- ページ全体を再読み込みしないのがポイント！

なぜJavaScriptを学ぶの？

- 現代のウェブ制作 (特にフロントエンド) に**不可欠**！
- HTML/CSSと一緒に学ぶ**基本スキル**です。
- 書いたコードで**ページが動く**のが楽しい！ 
- 学習情報 (本、Webサイト、動画) が**たくさん**あります。

まとめ

JavaScript は…

- HTML (骨組み) と CSS (見た目) でできたウェブサイト…
- 「動き」や「機能」を追加して…
- インタラクティブ（対話的）にするための…
- プログラミング言語です！

ウェブ開発の第一歩として、ぜひ触れてみましょう！

基本ルール①：文とセミコロン

- コンピューターへの一つ一つの指示を「**文** (ぶん)」と言います。
- 文の終わりには「**.**」(句点)の代わりに「**;**」(セミコロン)を付けるのが基本です。
 - 「ここまでが一つの指示ですよ」という目印。
 - 省略できる場合もありますが、付ける習慣をつけるのがおすすめです。

```
// 例：画面に「こんにちは」と表示する指示（文）  
console.log("こんにちは");
```

基本ルール②：コメント

- プログラムの中に、人間が読むための「メモ書き」を残せます。
- プログラムの実行には影響しません。
- `//` を書くと、その行の終わりまでがコメントになります。
- `/*` と `*/` で囲むと、複数行のコメントになります。

```
// これは一行コメントです  
let message = "Hello";
```

```
/*  
これは  
複数行の  
コメントです  
*/  
let count = 5;
```

基本ルール③：変数（データを入れる箱）

- 数値や文字などの「データ」を一時的に入れておくための「**名前付きの箱**」です。
- 後で名前を使って中身を出し入れできます。

変数の種類： `let` と `const`

- `let` (レット)
 - 中身を変えられる箱を作るときに使用します。
 - `let myScore = 80; myScore = 90; // OK`
- `const` (コンスト)
 - 中身を変えられない (固定の) 箱を作るときに使用します。
 - `const myName = "田中"; // myName = "佐藤"; はエラー`
- 使い分けのヒント
 - 基本は `const` を使う。
 - 後で値を変える必要がある場合だけ `let` を使う。
 - (`var` という古い方法もありますが、今は `let` `const` を使いましょう)

基本ルール④：データ型（箱に入れるもの）

変数（箱）には、色々な種類のデータを入れられます。

- **数値 (Number):** `100` , `3.14` など計算できる数字。
- **文字列 (String):** `"こんにちは"` , `'ABC'` など文字や文章。（`"` か `'` で囲む）
- **真偽値 (Boolean):** `true` (正しい) か `false` (間違い) のどちらか。
- **配列 (Array):** 複数のデータを順番に並べたリスト。 `["赤", "青", "黄"]`
- **オブジェクト (Object):** 名前と値のペアで情報をまとめたもの。 `{ name: "鈴木", age: 25 }`

データ型の例

```
let age = 30;           // 数値 (Number)
let message = "完了!";  // 文字列 (String)
let isActive = true;    // 真偽値 (Boolean)

let colors = ["赤", "青", "黄"]; // 配列 (Array)

let user = {            // オブジェクト (Object)
  name: "佐藤",
  score: 85
};
```


基本ルール⑤：演算子（計算や比較）

- 算術演算子: `+` (足し算), `-` (引き算), `*` (掛け算), `/` (割り算) など。
- 代入演算子: `=` (右の値を左の変数に入れる)。
- 比較演算子: `>` (より大きい), `<` (より小さい), `===` (完全に等しい) など。
 - 比べた結果は `true` か `false` になります。

```
let price = 100;  
let totalPrice = price * 1.1; // 100 * 1.1 = 110  
  
let score = 80;  
let isPass = (score >= 60); // scoreは60以上? -> true  
  
// 注意: 等しいか比べる時は `=` ではなく `===` を使う!  
let value = 10;  
console.log(value === 10); // true
```

基本ルール⑥：制御構造（もし～なら） - `if` 文

- 条件によって処理を変えたいときに使います。
- 「もし A なら B をする、そうでなければ C をする」といった流れを作れます。

```
let hour = 13;

if (hour < 12) {
  console.log("午前中です");
} else {
  console.log("午後です");
}
// この場合、「午後です」が表示される
```

基本ルール⑥：制御構造（繰り返し） - `for` 文

- 同じような処理を指定した回数だけ繰り返したいときに使います。

```
// 0から4まで、5回繰り返す
for (let i = 0; i < 5; i++) {
  console.log(i + "回目の処理");
}
```

```
/* 実行結果：
0回目の処理
1回目の処理
2回目の処理
3回目の処理
4回目の処理
*/
```

基本ルール⑦：関数（処理のまとめパック）

- よく使う一連の処理をまとめて、名前を付けたものです。
- 「レシピ」や「定型作業」のようなイメージ。
- 一度作れば、名前を呼ぶだけで何度でも使えて便利です。

```
// 「挨拶する」関数を定義
function greet(name) {
  console.log(name + "さん、こんにちは！");
}
```

```
// 関数を呼び出す（使う）
greet("佐藤"); // "佐藤さん、こんにちは！"
greet("鈴木"); // "鈴木さん、こんにちは！"
```

まとめ

- **文:** コンピューターへの指示。 `;` で区切る。
- **コメント:** 人間のためのメモ。 `//` や `/* */`。
- **変数:** データを入れる箱。 `let` (変えられる) `const` (変えられない)。
- **データ型:** 数値、文字列、真偽値、配列、オブジェクトなど。
- **演算子:** 計算 (`+`, `*`) や比較 (`===`, `>`)。
- **制御構造:** 条件分岐 (`if`) や繰り返し (`for`)。
- **関数:** 処理をまとめて再利用 (`function`)。

さあ、始めよう！

- 難しく考えず、まずは簡単なコードを書いて動かしてみましょう！
- ウェブブラウザの「開発者ツール」(F12キーで開くことが多い)の「コンソール」で、すぐに JavaScript を試せます。